Generación Automática de Interfaces a Partir de Patrones Estructurales de Tareas

José Ignacio Panach, Inés Pederiva, Sergio España, Óscar Pastor

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera s/n, 46071 Valencia, España
{jpanach, ipederiva, sergio.espana, opastor}@dsic.upv.es
+34 96 387 7000

Abstract. Actualmente, son muchos los modelos existentes para la captura de los requisitos funcionales de sistemas software. Sin embargo, la comunidad de Ingeniería del Software ha venido relegando a un segundo plano los requisitos de interacción, tratando la interfaz únicamente en tiempo de diseño. En este trabajo se extiende un sólido método de producción de software llamado OO-Method mediante la definición de una actividad de captura de requisitos de interacción. Para ello se hace uso de la notación formal CTT, una técnica bien conocida en el ámbito de la comunidad Interacción Persona-Ordenador. Se ha identificado un conjunto de patrones de interacción relevantes. Otorgando a estos patrones estructurales de tareas una sintaxis y una semántica bien precisas, se definen reglas de transformación entre el Modelo de Tareas y el Modelo de Presentación, el cual especifica de manera abstracta la interfaz. Esta aproximación se presenta instanciada para el entorno de la tecnología OlivaNova Model Execution, lo que permite la generación automática de la aplicación software.

1. Introducción

Los requisitos de interacción de un sistema software son una pieza clave en el proceso de desarrollo, pues de ellos parte el análisis y diseño de la interfaz de usuario. La interfaz de usuario permite el acceso y empleo de la funcionalidad soportada por la aplicación, por lo cual resulta un factor esencial para la aceptación del producto software por parte del usuario. Sin embargo, es a menudo ignorada por los procesos de desarrollo propuestos por la comunidad de Ingeniería del Software (IS), frecuentemente centrados en las entrañas funcionales del sistema. Por su parte, la comunidad de Interacción Persona-Ordenador (IPO) ha propuesto técnicas y notaciones adecuadas para la captura de requisitos de interacción y el modelado de la interfaz, si bien no suelen estar integradas en procesos que cubran todo el ciclo de vida del software, incluido modelado de datos y de procesos

La propuesta presentada en este trabajo integra técnicas IPO para la captura de requisitos de interacción en un sólido método de producción de software conforme con las directrices Model Driven Architecture (MDA) [6]. El estándar MDA define una estrategia de desarrollo basada en transformaciones entre modelos que describen el sistema software a distintos niveles de abstracción, de manera que se va aumentando

el detalle de especificación desde el modelo de requisitos hasta obtener finalmente el código fuente de la aplicación. Esta propuesta se ajusta de manera natural a las aproximaciones de la generación automática de código a partir de modelos conceptuales, entre las que se encuentra OO-Method [12] y su implementación industrial llamada OlivaNova Model Execution (ONME) [2]. OO-Method define un *Modelo Conceptual* que describe el sistema en el espacio del problema. Éste, a su vez, se compone de cuatro submodelos que son vistas con su correspondiente soporte gráfico para especificar mediante patrones abstractos los distintos aspectos del sistema de información: el *Modelo de Objetos* describe el dominio del negocio como un diagrama de clases, el *Modelo Dinámico* y el *Modelo Funcional* se centran en el comportamiento del sistema y constituyen su núcleo funcional y el *Modelo de Presentación* ofrece una descripción abstracta de la interfaz del sistema. El Modelo Conceptual, producido durante la fase de análisis, especifica la composición y el comportamiento del sistema con un alto nivel de abstracción. Mediante una estrategia de compilación de modelos se obtiene la aplicación que soporta el sistema de información especificado.

Para disponer de un ciclo de vida completo, se define una fase de captura de requisitos funcionales en la que participa el usuario, de modo que se facilita la identificación y especificación de sus necesidades. El resultado es un *Modelo de Requisitos Funcionales* [3] que se compone de la *Misión del Sistema*, una descripción de los objetivos de la organización, el *Árbol de Refinamiento de Funciones*, una estructuración de la funcionalidad de su sistema de información, y un *Modelo de Casos de Uso*, que describe con mayor detalle esta funcionalidad.

Pero la especificación de requisitos funcionales no es suficiente para obtener un modelo de requisitos completo. Es necesario añadir los mecanismos para la captura y especificación de los requisitos de interacción, y ese es el objetivo esencial aquí presentado. De esta manera, el analista de sistemas puede definir la interfaz del sistema junto con los usuarios. La estrategia presentada consiste en establecer una base de correspondencias entre un Modelo de Tareas y el Modelo de Presentación. Utilizando la notación CTT (*ConcurTaskTrees*) [13], bien conocida en ambientes HCI, se crea un *Modelo de Tareas* que describe los requisitos de interacción del usuario con el sistema que se pretende construir. Si este modelo cumple ciertas propiedades, es posible derivar una versión inicial del Modelo de Presentación, facilitando la tarea del analista.

La experiencia adquirida con esta aproximación [4] ha permitido reconocer dos aspectos susceptibles de mejora. En primer lugar, las correspondencias entre el Modelo de Tareas y el Modelo de Presentación deben relacionar patrones estructurales de tareas con sus correspondientes patrones de interfaz. Para esto es necesario definir detalladamente cada configuración de tareas que constituye un patrón de interacción relevante, insistiendo en el aspecto estructural. Con ello, se puede aportar una consistencia gramatical al Modelo de Tareas, lo que posibilita la posterior derivación del Modelo de Presentación.

En segundo lugar, la experiencia en los proyectos en los que se ha aplicado la estrategia propuesta ha puesto de relevancia la dificultad que entraña tratar con la notación CTT. El gran tamaño de los diagramas de tareas cuando se emplean para modelar la interacción con sistemas de información organizacionales los hace prácticamente inmanejables, aún disponiendo de herramientas para su edición [9][10]. En trabajos anteriores de los autores [4], se propuso una derivación de los CTT a través de Casos de Uso. En el trabajo actual se presenta otro tipo de derivación. La idea es colocar una

capa de "*sketch"* por encima del modelo de tareas para facilitar esa captura y validación de los requisitos con el usuario (ver Figura 1). En los casos en que la organización cuya gestión está siendo analizada tiene conocimientos informáticos, los usuarios que participan en la captura de requisitos se encuentran más cómodos hablando en términos de la interfaz que necesitan. El analista realizaría bocetos de la interfaz junto con el usuario y éstos serían interpretados por una herramienta de reconocimiento que detecte los elementos de la interfaz que se están bosquejando, en la línea de otras herramientas para este propósito [3]. En paralelo y en tiempo real, la herramienta iría construyendo el modelo de tareas subyacente a los bocetos ("*sketches")* de la interfaz, de manera que se dispondría de una base formal a partir de la cual se puede verificar la consistencia de los requisitos capturados. Para que esto sea posible, es necesario determinar con precisión un conjunto de patrones de interacción, que serán los utilizados como unidades básicas de construcción del Modelo de Tareas. Este enfoque constituye otra aportación relevante de este trabajo.

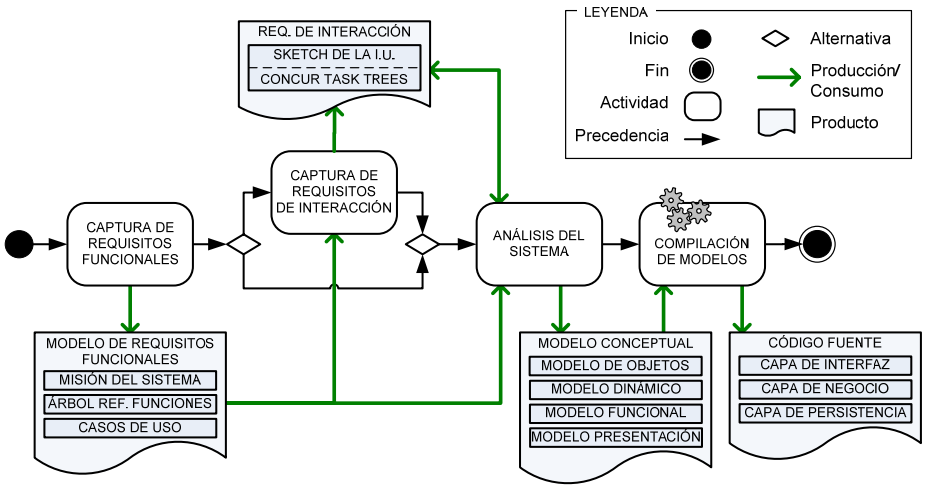

Fig. 1. Inclusión del modelado de requisitos de interacción en el proceso OO-Method.

Para alcanzar esos objetivos, la estructura del artículo es la siguiente: en la sección 2 se recogen algunas aproximaciones que han definido correspondencias entre los modelos de tareas y los modelos de interfaz. En la sección 3 se explica el Modelo de Presentación y los patrones que lo constituyen. La sección 4 presenta con detalle varios patrones estructurales de tareas que hemos identificado y sus correspondencias con los patrones del Modelo de Presentación. Por último, la sección 5 resume algunas conclusiones interesantes y los retos planteados por los trabajos futuros.

2. Estado del arte

Existen trabajos abordados con el uso de CTTs para expresar, en una notación formal, la interacción persona-ordenador como una descomposición de las tareas en forma gráfica. En UsiXML [8] se hace uso de la notación CTT para validar la interacción

con el usuario y como una herramienta para detectar patrones de interacción. Este modelo, al que denominan Modelo de Tareas, es utilizado para hacer una vinculación con el Modelo de Interfaz Abstracto y con el Modelo de Dominio, tanto en forma automática como manual.

Por otro lado, Wisdom [11] propone una extensión a la notación CTT, basada en UML, a la cual llaman Modelo de Diálogo; este modelo se utiliza para validar la interfaz. También propone un Modelo de Presentación Abstracto donde se especifica la estructura de las componentes de interfaz.

SUIDT (*Safe User Interface Design Tool)* [1] es una herramienta que genera interfaces automáticamente usando varios modelos relacionados entre sí: Núcleo Formal Funcional, Modelo Abstracto de Tareas y Modelo Concreto de Tareas. Con el *Núcleo Formal*, se modela el funcionamiento de la aplicación usando seudo-formalismos que se corresponden con pre- y post-condiciones. Con el *Modelo Abstracto de Tareas* se definen las necesidades del usuario en términos de tareas, utilizando para ello la notación CTT. Este modelo debe ser independiente de la interfaz a desarrollar. Por último, el *Modelo Concreto de Tareas* describe las acciones de la aplicación y la interacción del usuario con la interfaz. Para ello utiliza CTT enriquecido con nuevas categorías usadas para modelar la interacción. SUIDT solo genera la interfaz de la aplicación final, no implementa la funcionalidad. La especificación funcional solo se utiliza para asegurar que la interfaz respeta las reglas de comportamiento impuestas por el usuario.

Analizados de forma global, estos trabajos no construyen la interfaz a partir de patrones de interacción predefinidos, por lo que no es necesario construir un modelo de tareas con restricciones; sin embargo esto impide la definición de transformaciones cerradas que confluyan en la generación completa de la interfaz y de la aplicación final totalmente funcional.

Por otro lado, el diseño de CTT en UsiXML y en SUIDT no es presentado dentro de un proceso de desarrollo completo y en todos los casos es tratado en un mismo nivel de abstracción en el que se presentan los modelos de objetos y de interfaz abstracta, sin usarlo como una herramienta de un nivel superior con la cual validar tempranamente los requisitos.

3. Modelo de Presentación

De los modelos que define OO-Method, en este trabajo nos centraremos en el Modelo de Presentación, que permite la especificación abstracta de las interfaces de usuario por medio de un lenguaje de patrones [7]. El modelo se estructura en tres niveles o capas de especificación:

- **Nivel 1. Árbol de Jerarquía de Acciones (AJA):** expresa cómo la funcionalidad será presentada al usuario que acceda al sistema.
- **Nivel 2. Unidades de Interacción (UIs):** modela los escenarios básicos de interfaz que el usuario deberá emplear para llevar a cabo sus tareas. Existen cuatro tipos: (1) UI de servicio: modela la presentación de un diálogo cuyo objetivo es que un usuario lance un servicio; (2) UI de instancia: modela la presentación de los datos o estado de una instancia; (3) UI de población: su objetivo es mostrar un conjunto de

instancias para una clase dada; (4) UI de maestro/detalle: como ejemplo de UI compleja, que se construye a partir de unidades de interacción más sencillas.

- **Nivel 3. Patrones Elementales (PEs):** permiten restringir y precisar el comportamiento de las diferentes unidades de interacción. Son ejemplos de patrones elementales: (1) Filtro: permite hacer un filtrado de información a partir de un criterio de filtrado; (2) Criterio de ordenación: proporciona un mecanismo de ordenación de objetos basado en las propiedades de estos; (3) Conjunto de visualización: describe una lista de propiedades observables (atributos) de los objetos ; (4) Acciones: determina las acciones que son ofrecidas a un usuario tras seleccionar un objeto; (5) Navegación: determina qué información relacionada con el objeto actual es alcanzable en un contexto dado.

4. Correspondencias entre CTT y el Modelo de Presentación

Tal como aparece en la Figura 1, los requisitos de interacción van a ser capturados a través de técnicas de "*sketching*" o maquetado. Los resultados obtenidos con estas técnicas van a estar soportados por un Modelo de Tareas con notación CTT [13] de manera transparente para el analista.

Este trabajo está centrado en el Modelo de Tareas, cuya gramática asociada tiene los siguientes componentes:

- Léxico: viene dado por propia notación CTT (con sus tipos de tareas de interacción, de sistema y abstractas).
- Sintáctico: lo conforman los patrones estructurales de tareas, que son estructuras de tareas relacionadas por operadores temporales, a los que añadimos un mecanismo de composición de patrones.
- Semántico: viene dado por la correspondencia entre los patrones de tareas y los patrones del Modelo de Presentación.

Los patrones estructurales de tareas han sido definidos de manera genérica y, por lo tanto, presentan argumentos que deberán ser instanciados cuando el patrón sea usado para modelar una interfaz determinada. En las figuras siguientes, estos argumentos se muestran en letra cursiva, indicando que su nombre debe ser instanciado. Los argumentos de cardinalidad variable se representan con puntos suspensivos (p.e. 1..N)

A continuación se presentan los patrones estructurales de tareas correspondientes a los siguientes patrones del Modelo de Presentación: UI de Población y patrones elementales relacionados.

- **Filtro:**

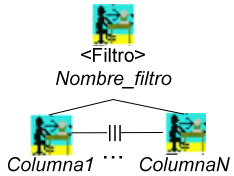

Fig. 2. Filtro con notación CTT

Obsérvese que el patrón CTT propuesto presenta varios argumentos: por un lado el nombre del filtro y por otro los campos que definen el filtro.

- **Criterio de ordenación:**

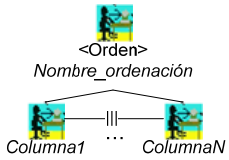

Fig. 3. Criterio de ordenación con notación CTT

En la Figura 3 cada tarea de interacción que es una hoja representa cada uno de los criterios de ordenación posibles.

- **Acciones:**

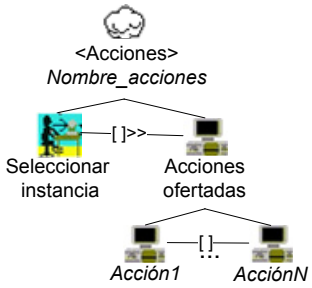

Fig. 4. Acciones con notación CTT

Tal como aparece en la Figura 4, la tarea de interacción modela la selección de la instancia sobre la cual se aplicarán las acciones que se modelan con el resto de tareas de sistema. Estas acciones son funciones del negocio y por eso son argumentos.

- **Navegación:**

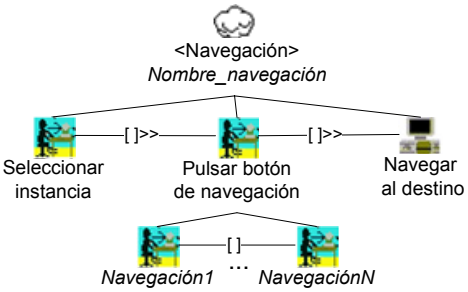

Fig. 5. Navegación con notación CTT

En la Figura 5 se representa, mediante tareas de interacción, la selección de la instancia desde la cual se va a navegar y la información hacia la cual se desea navegar. El conjunto de los posibles destinos constituye los argumentos de este patrón. Después, la tarea de sistema es la encargada de realizar la navegación.

- **Conjunto de visualización:**

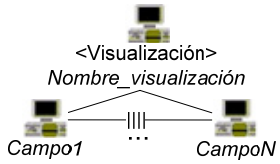

Fig. 6. Conjunto de visualización con notación CTT

La Figura 6 muestra, mediante tareas del sistema, los campos a visualizar. La instanciación de estos argumentos es determinada por las necesidades de consulta de la interacción modelada.

Una vez definidas las correspondencias entre estructuras de tareas y patrones de tercer nivel del Modelo de Presentación, podemos realizar la misma labor con el patrón de Unidad de Interacción de Población, que integra los patrones de tercer nivel mencionados con anterioridad.

Tal como muestra la Figura 7, el árbol que representa el patrón de población incluye tareas de interacción que se encargan de realizar el filtrado (*Filtro*) y la ordenación (*Orden*). Los dobles corchetes representan las reglas de composición gramatical, puntos donde se enganchan otros patrones estructurales de tareas. Posteriormente, existe una tarea de sistema que se encarga de mostrar las instancias de los objetos por pantalla (*Visualización*), filtradas y ordenadas por los criterios seleccionados. Por último, el usuario puede efectuar operaciones de *Acción* y *Navegación* con estas instancias, representadas en el diagrama mediante tareas abstractas.

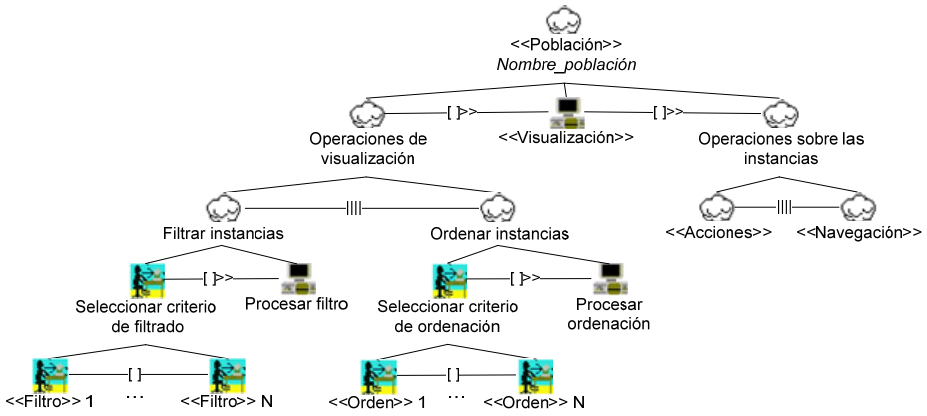

Fig. 7. Interfaz de usuario de población

Una vez construidos los CTT utilizando los patrones estructurales de tareas definidos, se aplican las reglas de transformación para obtener el Modelo de Presentación. Al añadir el resto de modelos de OO-Method (Modelo de Objetos, Dinámico y Funcional) y aplicar la compilación de modelos, se obtiene de manera automática la interfaz final del sistema con toda su funcionalidad, tal como muestra la Figura 1.

5. Caso de Estudio

Para ilustrar con un ejemplo práctico la propuesta, se ha elegido el sistema software de una empresa que se encarga de suministrar agua. Este sistema se encarga de gestionar las lecturas de los contadores, emitir facturas, registrar el uso de materiales en las reparaciones del sistema de reparto de agua y el mantenimiento de materiales en los almacenes. Se presenta la tarea **Listar contadores** a modo de ejemplo. En esta tarea se listan todos los contadores registrados en el sistema.

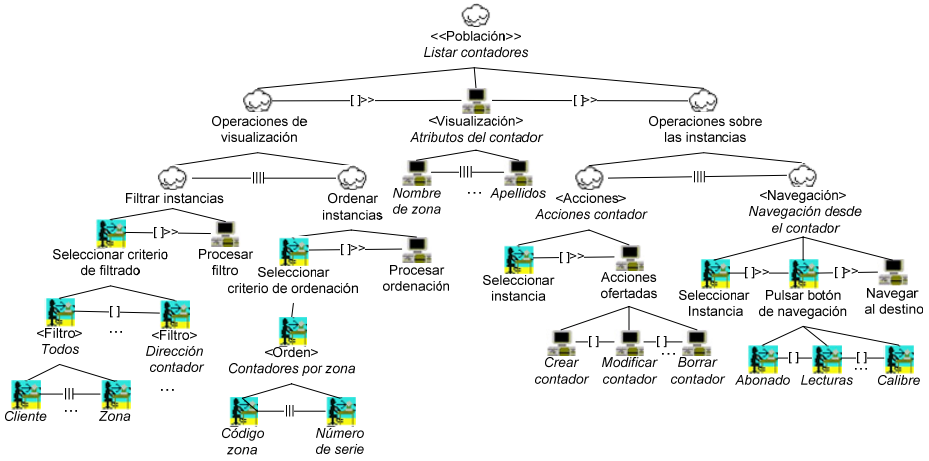

Fig. 8. CTT para Listar contadores

En la Figura 8 se muestra el CTT construido mediante los patrones estructurales de tareas. Cada uno de los nodos del árbol CTT que están en cursiva, son parámetros que han sido instanciados durante la captura de requisitos de interfaz. Se puede observar qua la tarea seleccionada contiene todos los patrones estructurales que se pueden representar en un CTT que modele una interfaz de población: varios filtros, un criterio de ordenación, un conjunto de visualización, acciones y posibles navegaciones. Al aplicar las correspondencias entre los patrones estructurales de tareas y los patrones del Modelo de Presentación, explicadas en el capítulo anterior, se obtienen de manera automática los patrones de presentación (ver Figura 9).

En la Figura 9, se destacan en blanco los patrones del Modelo de Presentación correspondientes a Listar contadores, y se muestran en gris otros patrones creados a partir de otros árboles CTT del Modelo de Tareas. A partir del Modelo de Presentación la tecnología ONME genera automáticamente la interfaz final.

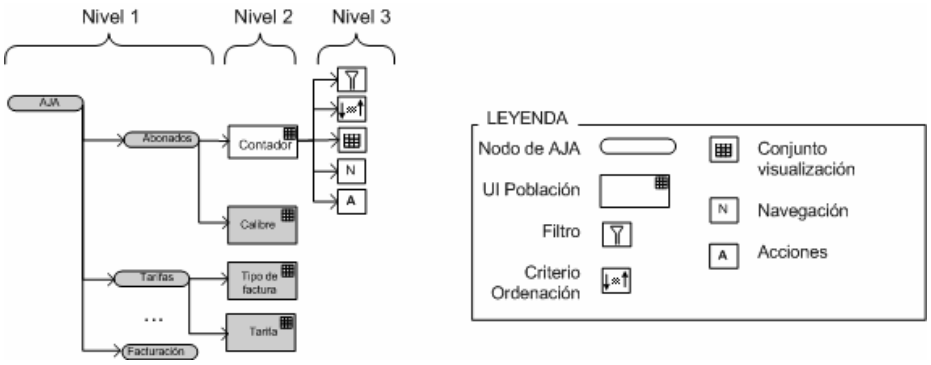

Fig. 9. Modelo de Presentación

Tal como se deduce de la Figura 10 cada patrón estructural del diagrama CTT, ha dado lugar a un patrón de tercer nivel del Modelo de Presentación (cuyas componentes de interfaz aparecen marcadas con elipses negras en la figura). Al aplicar la compilación de modelos se genera la interfaz mostrada en la figura.

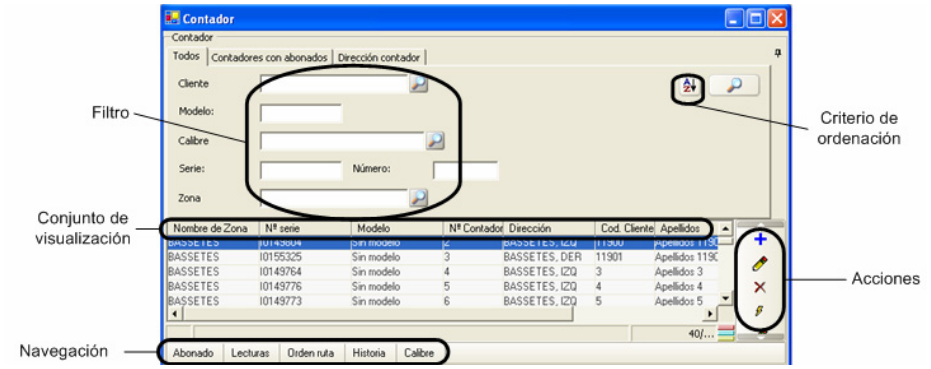

Fig. 10. Pantalla para listar los contadores

5. Conclusiones y trabajos futuros

En este trabajo se ha presentado una técnica de especificación de requisitos de interacción basada en construir un Modelo de Tareas usando patrones estructurales de tareas, a partir de los cuales se puede derivar automáticamente el Modelo de Presentación. Esta técnica mejora el proceso de desarrollo que, soportado por la tecnología ONME, permite generar automáticamente el código de la aplicación a partir de los modelos de Presentación, de Objetos, Dinámico y Funcional.

La experiencia indica que los diagramas CTT pueden llegar a convertirse en ilegibles para aplicaciones medianas. Es por ello, que el proceso de generación de los CTT debe ser oculto para el analista. Se ha propuesto como solución el uso de bocetos ("*sketches*") para generar dichos diagramas. Como las reglas de construcción de los

CTT han quedado perfectamente definidas, el analista está limitado a utilizar las reglas existentes. Esta limitación afectará a la construcción de los bocetos pero, determinando de forma precisa los gránulos de especificación, se puede garantizar la construcción del boceto y, en paralelo, del Modelo de Tareas que permitirá derivar el Modelo de Presentación.

Como trabajo futuro se plantea estudiar el conjunto de constructores para el "*sketch*" y su correspondencia inyectiva con un patrón estructural de tareas. Esta propuesta se completaría con la implementación de una herramienta que, a partir de los bosquejos, construya paralelamente y de forma transparente para el analista los CTT. Esta herramienta también deberá implementar las transformaciones de patrones estructurales de tareas a patrones del Modelo de Presentación descritas en este trabajo.

Referencias

[1] Baron M., G. P. (Romania 2002). "SUIDT: A task model based GUI-Builder." Task MOdels and DIAgrams for user interface design (TAMODIA): 64-71.

[2] Care Technologies: http://www.care-t.com Última visita: Nov-2005.

[3] Coyette, A., Vanderdonckt, J., (2005) A Sketching Tool for Designing Anyuser, Anyplatform, Anywhere User Interfaces, Proc. of 10th IFIP TC 13 Int. Conf. on Human-Computer Interaction Interact'2005 (Rome, 12-16 September 2005), M.-F. Costabile, F. Paternò (eds.), LNCS, Vol. 3585, Springer-Verlag, Berlin, pp. 550-564.

[4] España, S., Pederiva, I., Panach, J. I. (2006) Integrating Model-based and Task-based approaches to user interface generation. Demo paper in Proc. of Conference on Computer-Aided Design of User Interfaces (CADUI 2006), Romania (en impresión)

[5] Insfrán, E., Pastor, O., Wieringa, R. (2002). Requirements Engineering-Based Conceptual Modelling. Requirements Engineering, Vol. 7, Issue 2, p. 61-72. Springer-Verlag.

[6] MDA "Model Driven Architecture" http://www.omg.org/mda Última visita: Nov-2005.

[7] Molina, P. J. (2003). Especificación de interfaz de usuario: de los requisitos a la generación automática. PhD. Departamento de Sistemas Informáticos y Computación. Valencia, Universidad Politécnica de Valencia: 382.

[8] Montero, F., V. López-Jaquero, et al. (2005). Solving the mapping problem in user interface design by seamless integration in IdealXML. Proc. of DSV-IS'05, Newcastle upon Tyne, United Kingdom, Lecture Notes in Computer Science, Springer-Verlag, Berlin.

[9] Mori G., Paternò F., Santoro C. (2002) "CTTE: Support for Developing and Analyzing Task Models for Interactive System Design" IEEE Trans. on Software Engin.; pp.797-813.

[10] Mori G., Paternò F., Santoro C. (2004) "Design and Development of Multidevice User Interfaces through Multiple LogicalDescriptions" IEEE Transactions on Software Engineering; pp.507-520.

[11] Nunes, N. J. y J. F. e. Cunha (2000). "Wisdom: a software engineering method for small software development companies." Software, IEEE 17(5): 113-119.

[12] Pastor, O., Gómez, J., Insfrán, E. Pelechano, V. (2001) The OO-Method Approach for Information Systems Modelling: From Object-Oriented Conceptual Modeling to Automated Programming. Information Systems, 26(7) 507–534.

[13] Paternò, F., C. Mancini, et al. (1997). ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models. Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction, Chapman & Hall, Ltd.: 362-369.

[14] Molina J.C., Pastor O. "MDA, OO-Method y la tecnología OlivaNova Model Excecution". I Taller sobre desarrollos dirigidos por modelos, MDA y aplicaciones. Málaga, 2004.